

Homework 4

February 26, 2008

Due Date: March 4

You should not give any code or even any pseudo-code in this homework. If there is any uncertainty about how to describe a solution in English, please ask us. The TAs will not look at any code or pseudo-code that is written. It is important to learn to describe a design for a data structure or high-level description of an algorithm in English.

1. (25 points) In this problem we consider a collection of elements that correspond to cell phones that are in the range covered by a given tower. For each cell phone object the following instance variables are stored:
 - number (10 digit number)
 - latitude
 - longitude
 - name
 - total time in use (while in range of the tower)

We consider a variety of applications that all depend upon this data. All should allow new elements to be added or removed from the collection as cell phones go in and out of range of the tower.

For each of these applications indicate which untagged algorithmically positioned collection ADT would be the best fit and which instance variable(s) should be used to organize the data. For example, if you were going to use a comparator, which instance variable(s) should be used to define the comparator. You should give a brief explanation as to why the ADT you chose is the best fit for the problem.

- (a) (5 points) Given a phone number return the name of the person with that phone number, and their location.
 - (b) (5 points) For a given location with latitude x and longitude y and range d , return the list of all people whose cell phones have a latitude between $x - d$ and $x + d$ (inclusive), and a longitude between $y - d$ and $y + d$ (inclusive)
 - (c) (5 points) Iterate through the list of the names in alphabetical order of all users with a cell phone in the collection.
 - (d) (5 points) Return a sorted list of all phone numbers in the collection that begin with a specified prefix. For example, the application might want to receive all cell phones with area code 314.
 - (e) (5 points) Return the cell phone number that has had the most usage time.
2. (15 points) You are given a set S of w words to place in a collection of words to be used by a spell checker. You are also given an n word document. Your task is to list all words in the document that are not in S . Give an $O(n + w)$ expected time algorithm that only requires space for at most $2w$ references (plus the space needed to store the words in S and the words in the document).

Be sure to analyze the time and space complexity of your solution giving enough detail to argue that your solution satisfies the stated goals.

3. (25 points) This problem focuses on ADT selection. You do not need to select specific data structures. Your solution should be at most one page. It is important that you learn how to concisely describe your solution to this type of a problem. Your solution should provide the information to the following questions.

- Describe your ADT choice(s) including all decisions such as whether it is a collection, tagged collection (specifying the tag), or tagged bucket collection (specifying the tag and bucket type).
- Briefly describe how to perform each requested operation. Do NOT give an code or even pseudo-code. Just describe which ADT methods are used.
- Briefly overview the time efficiency for each operation. You need not perform any analysis of the methods used – just state what it is.

You are to design a simple key frame animation system where n is the number of frames stored in your data structure. You are to support the following methods.

- `insertFrame(frameID, frame, timeInMovie, artist)` where `frameID` is a unique id for the frame, `frame` is an image, `timeInMovie` is the offset in seconds that the frame will be shown in the movie, and `artist` is the name of the artist who designed the frame. You should assume that the frames are not inserted in the order in which they appear in the movie. This method should run in worst-case or expected-case $O(\log n)$ time.
- `produceMovieSegment(startTime, endTime)` produces a list of all frames, sorted by time, with times between `startTime` and `endTime`, inclusive. This should run in worst-case or expected-case $O(k + \log n)$ time where k is the number of frames in the given time range.
- `getFrames(artist)` should return an iterator for the frameIDs (in any order) for all frames produced by the given artist. You can assume the artists have unique names. This method must run in worst-case or expected-case $O(1)$ time.

4. (25 points) For the following problem you are to select the data structure(s) to use. The following components should be clearly given in your solutions. Your solution should be at most one page.

- You should very clearly describe your data structure choice, including all decisions about how the data structure is to be applied (e.g., what is used as the tag, what is the associated data, what is the table size and collision resolution choice for each hash table,...).
- You should clearly describe how each of the provided operations will be implemented AND discuss the efficiency for each of the operations. You only need to describe any new methods or variations of methods covered in class that you need. You can specify any standard method you are using (e.g., insertion) along with its parameters (e.g., what is given as the tag, and what is given as the associated data), and state its time complexity.
- For any Set ADT data structure used be sure to specify the hash table size and the method for collision resolution.

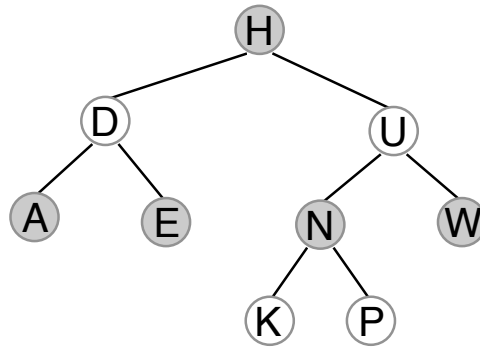
You have been hired as a consultant by University X that has an enrollment of s students where s is approximately 10,000. Each student has a unique 6 digit id number. For each student there is a `StudentRecord` that includes a name, an email address, a mailing address, current courses (initially empty) and a transcript. Each course also has unique id (e.g. E81241FL07), and an associated `CourseRecord` that includes a professor, classroom and class list (initially empty). You can assume that there are t courses where t is roughly 1000. The size of course i is denoted by n_i where $0 \leq n_i \leq 300$. Finally, you can assume that each student takes no more than 10 courses in a semester.

You should treat s , t , and n_i (for $1 \leq i \leq t$) as variables in analyzing and stating the asymptotic time complexity for the methods.

You are to design a solution to implement the following operations in an efficient manner. There are some trade-offs which you'll need to think about and decide which option is best. There is not a single right answer. You just need to give a justification for your choices.

- `insertStudent(int studentID, StudentRecord data)` inserts the given student. You should assume that this method is called somewhat frequently.
- `removeStudent(int studentID)` removes the student with the given id. You should assume that this method is used infrequently.
- `insertCourse(int courseID, CourseRecord data)` inserts a new course. This method will be used relatively infrequently. Note that courses are never removed from the system.
- `registerStudent(int studentID, int courseID)` registers the given student into the given course. You should assume that this method is called very frequently.
- `withdrawStudent(int studentID, int courseID)` removes the given student from the given course. You can assume that this method is called relatively frequently.
- `printSortedRoster(int courseID)` returns an alphabetized list (or an iterator over an ordered collection) of all students in the course. You can assume that this method is called extremely often and should be as efficient as possible.

5. (10 points) Insert C and then L into the following red-black tree where the filled nodes are black and the unfilled nodes are red. You should show the red-black tree that results after each rotation. To help us give you partial credit if you make errors, please show enough intermediate steps that we can follow your work.



Note: We will cover the red-black tree in class on Thursday. So you should wait until after that class to answer question 5.

Extra Credit Problem:

6. (3 points) Prove that any comparison-based algorithm for constructing a binary search tree from an arbitrary list of n elements takes $\Omega(n \log n)$ time in the worst case.

Hint: Think about reducing the problem of sorting to performing a set of operations on a binary search tree. That is, show that if a faster algorithm existed for constructing a binary search tree then you would violate the $\Omega(n \log n)$ comparison-based sorting lower bound.